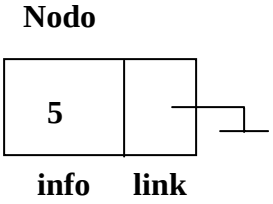
U T E M
Facultad de Ingeniería
Departamento de Informática y Computación

Programación INF627
2010

Nodo

Es una estructura que permite almacenar información.



Implementación de un nodo

```
typedef struct Nodo{
      int info;
      struct Nodo *link;
};

typedef Nodo *Lista;
```

Observación:

- La estructura NODO permite almacenar un dato (int) y un puntero de tipo NODO
- El tipo LISTA es un puntero a una estructura de tipo NODO

```
int main()
{
  Lista p;
  p = (Lista) malloc(sizeof(Nodo));
  p->info = 5;
  p->link = NULL;

  system("pause");
}
```

Nota:

- Se define la variable p de tipo Lista, es decir, p es un puntero que "apunta" a una estructura de tipo Nodo.
- Crea un espacio de memoria de tipo Nodo y lo apunta por p
- Asigna el valor 5 en el campo info del nodo apuntado por p
- Asigna NULL en el campo link del nodo apuntado por p

Observación:
NULL es una constante, que está definida como 0 en C. NULL es una dirección nula (dirección inexistente) que nos sirve para indicar puntero vacio o fin de lista.

Ejemplo 1

```
int main()
{
  Lista p;
  p = (Lista) malloc(sizeof(Nodo));
  p->info = 5;
  p->link = NULL;

  p = (Lista) malloc(sizeof(Nodo));
  p->info = 8;
  p->link = NULL;

  system("pause");
}
```

Obs. Hay que tener cuidado cuando se crean nodos. En el ejemplo se crea un nodo apuntado por p y se asigna el valor 5; luego se crea otro nodo apuntado por p y se asigna el valor 8. Al finalizar el código se tiene en memoria dos nodos pero sólo se tiene acceso al nodo apuntado por p (valor 8), el nodo con el valor 5 sigue existiendo pero no tenemos acceso a el.

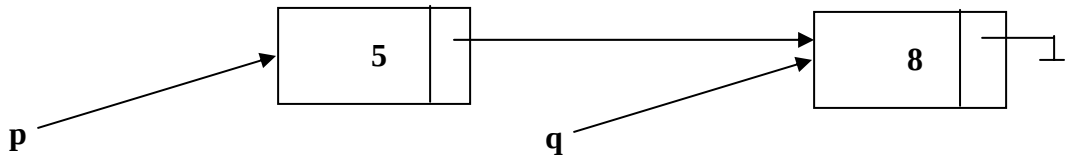
Ejemplo 2

```
int main()
{
  Lista p, q;
  p = (Lista) malloc(sizeof(Nodo));
  p->info = 5;
  p->link = NULL;

  q = (Lista) malloc(sizeof(Nodo));
  q->info = 8;
  p->link = NULL;

  p->link = q;  //permite enlazar el nodo 5 con el nodo 8

  system("pause");
}
```



Obs. Se puede observar que ambos nodos quedan enlazados.

Ejemplo 3

```
int main()
{
  int i, n;
  Lista L, p, q;
  L = NULL;

  printf("Ingrese cantidad de nodos : ");
  scanf("%d",&n);
  //permite crear N nodos
  for(i=1; i<=n; i++)
  {
    p = (Lista) malloc(sizeof(Nodo));
    p->info = i;
    p->link = L;

    L = p;
  }
  //permite recorrer y mostrar el contenido de una lista
  q = L;
  while(q != NULL)
  {
      printf("Valor : %d\n",q->info);
      q = q->link;
  }
}
```

Observaciones:

- Se define un puntero L y se inicializa en nulo, para indicar el inicio de la lista vacía
- Se crean “n” nodos y se agregan al inicio de la lista L
- El puntero L siempre queda al inicio de la lista
- Se recorre la lista, con un puntero auxiliar q. Se avanza a través de la lista con q=q->link

Ejemplo 4 . Uso de funciones

```
void imprimir(Lista L)
{
  Lista q;
  q = L;
  while(q != NULL)
  {
      printf("Valor : %d\n",q->info);
      q = q->link;
  }
}
```

```
void agregarInicio(Lista *L, int e)
{   Lista p;
    p = (Lista) malloc(sizeof(Nodo));
    p->info = e;
    p->link = *L;

    *L = p;
}
int main()
{
  int i, n;
  Lista L, p, q;
  L = NULL;
  printf("Ingrese cantidad de nodos : ");
  scanf("%d",&n);
  //permite crear N nodos
  for(i=1; i<=n; i++)
  {
    agregarInicio(&L, i);
  }
  imprimir(L);
}
```

Observaciones:

- Procedimiento **agregarInicio(L, e)**, permite crear un nodo con información e, y lo enlaza al inicio de la lista L.
- Procedimiento **imprimir(L)**, permite recorrer e imprimir el contenido de todos los nodos de la lista L.

Ejemplo 5

```
void agregarFinal(Lista *L, int e)
{
    Lista p, q;
    p = (Lista) malloc(sizeof(Nodo));
    p->info = e;
    p->link = NULL;
    if(*L == NULL)
         *L = p;
    else
    {
       q = *L;
       while(q->link != NULL)
       {
         q = q->link;
       }
       q->link = p;
    }
}
```

Ejercicios Propuestos

Escribir los siguientes operadores sobre listas:

1. **Contar(L)**, retorna la cantidad de nodos de la lista L
2. **Sumar(L)**, retorna la suma de los nodos de la lista L
3. **Mayor(L)**, retorna la el mayor valor entre los nodos de la lista L
4. **ContarSI(L, k)**, retorna la cantidad de nodos que contienen elementos que comienzan con el dígito k, en una lista L.
5. **AgregarKfinal(L, k, e)**, agrega k nodos con información “e” al final de la lista L.
6. **EliminarPrimero(L)**, permite eliminar el primer nodo de la lista L
7. **EliminarUltimo(L)**, permite eliminar el último nodo de la lista L
8. **EstaOrdenado(L)**, Determina si se encuentran o no ordenados los elementos contenidos en la lista L.
9. **Insertar(L, e)**, permite insertar un nodo con información e, manteniendo el orden de la lista L.
10. **Eliminar(L, e)**, permita eliminar el nodo que contiene información e en una lista L.